2013.4.28 上瀧　剛
http://d.hatena.ne.jp/suzume_r/

クマ将棋

熊本から来ましたのでクマ将棋といいます。穴熊とは無関係です。
今回で2回目の参加です。

【前回からの変更点】
①探索の強化
前回：Bonanza6.0のfv.binを使った場合で、floodgateでR2200 (CPU1コア)
今回：上記の条件でR2600 (CPU1コア)
※大会では自前の評価関数を使う。現在、R2000程度?

指手のオーダリングはbonanza風
futility・reduction・静止探索はstockfish風

②指手生成の高速化
ビットボードや利き情報を持たないが高速。
前回：指手生成祭ベンチマーク局面で65万回/秒
今回：300万回/秒
※利き生成が苦手なためSEEや一手詰めも独自の工夫をしています。

③学習方法の変更
オンライン学習→3日程度で学習終了

指手生成の高速化

ビットボードや利きテーブル不要で高速な指手の生成
・考えられる移動パターンをswitch・if文で展開
（指手を生成するコードを生成するコードを作成）
・事前に移動判定用のテーブル等が不要（盤面データを関数に渡すだけでよい）
・局面構造はシンプル

```
int *gen_evasion(char *kifu, int *te, char king_pos, char attack_pos)
{
        switch(king_pos)
        {
        case 0: // kpos
                switch(attack_pos)
                {
                case 9: // attack pos
                        switch(kifu[1])
                        {
                        case KOMA_GI:
                                *te++=0x24089;
                                *te++=0x20089;
                                break;
                        case KOMA_KA:
                                *te++=0x34089;
                                *te++=0x30089;
```

王手回避手生成のコード例（700万行あります）

```
// 局面のデータ構造
struct kyokumen_t
{
    // 盤面＋持ち駒（歩香桂銀金角飛）
    char kifu[81 + 14];
    // 王の位置
    char king_pos[2];
    // ハッシュキー(盤面と持ち駒)
    uint64 hash_ban;
    uint64 hash_hand;
    // 駒割評価
    int material;
};
```

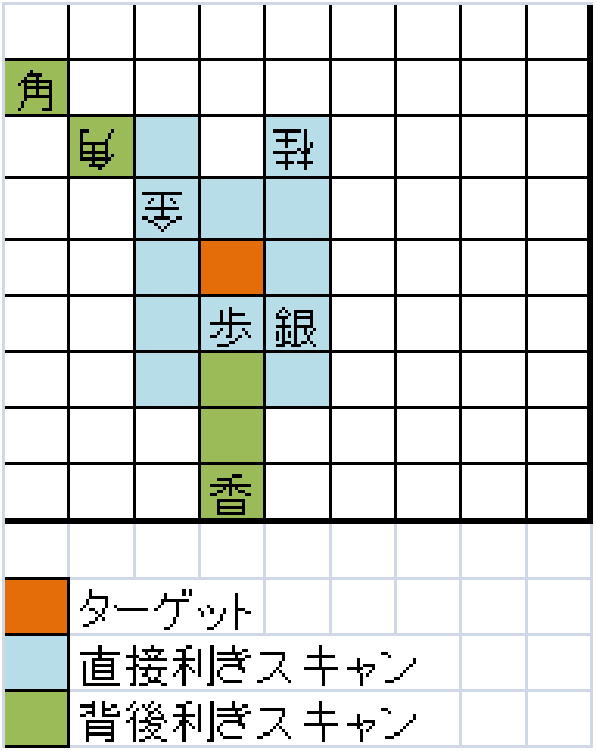
基本局面構造

指手生成祭ベンチマーク局面で300万回/秒となり、
自作のmagic bitboard版と遜色ない速度を達成

SEE(駒の交換値)の計算方法

問題点　ビットボードでないため利きの計算が不得意→SEE計算が遅い

対策　　盤配列スキャンによる駒交換リスト作成。これを使ってSEE計算。



※跳び利きスキャンもあり

ターゲット周囲のマスをスキャンして以下のリスト作成

	直接利き	背後利き	背後利き
LIST1	▲歩	▲香	
LIST2	▲銀		
LIST3	▽金	▽角	▲角
LIST4	▽桂		

▲歩で取ったら
背後の利きでリスト更新

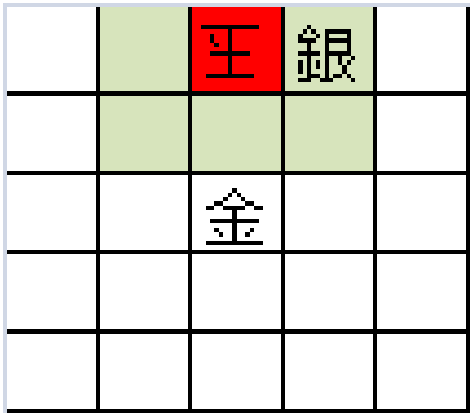
	直接利き	背後利き	背後利き
LIST1	▲香		
LIST2	▲銀		
LIST3	▽金	▽角	▲角
LIST4	▽桂		

指手生成コードと同様に、switch文を展開して、高速にリストを生成可能

高速一手詰ルーチン

問題点　ビットボードでないため利きの計算が不得意→王手の生成が苦手

対策　　盤配列スキャンによる利きリスト作成。これを使って簡易一手詰めを計算。



玉周辺のマスをスキャンして、
玉周辺３×３マスに対して以下の情報を取得

①先手がそこに移動できる手
②先手が王手となる手（移動、駒打）
③後手がそこに移動できる手
④空白か否か

リストから駒打ち王手、移動王手、玉の退避判定を行う。

指手生成コードと同様に、switch文を展開して、高速にリストを生成可能。
桂馬の成り王手も生成できる。70万回/秒程度の速度。

オンライン学習

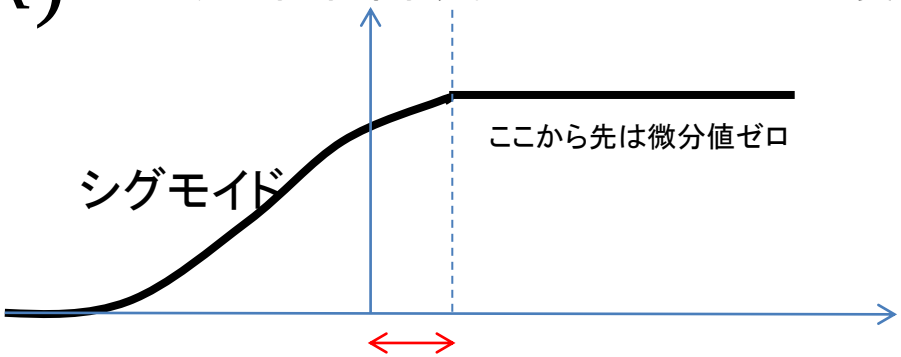
開発効率化のため、オンライン学習を採用。学習の偏りを減らすために
学習局面をランダムにシャッフル。各局面で指手を30手ランダムサンプリング。
過学習を抑えるために正則化およびマージン打ち切り（正解した場合は更新しない）を採用。
末端局面で王手状態の場合は、重みを大きめに更新→玉の危険度を優先的に評価

$$E(\mathbf{w}_t) = \sum_{\text{局面}m} \sum_{\text{指手}k=1}^{30} l\left(\mathbf{w}_t \cdot B_{m,0} - \mathbf{w}_t \cdot B_{m,k}\right) + \lambda \left|\mathbf{w}_t\right|$$

過学習を抑えるための
L1正則化項

$B_{m,k}$ ：局面mにおいてk番目の指手を探索を進めていったときの末端局面

$l(x)$ ：一致率評価関数。シグモイド＋進行度に合わせたマージン打ち切り



マージン：終盤ほど大きくする。

数日で学習が終わる。ただし、思ったような成果（特に終盤が弱い）は出ていません。